CPSC 1155 – Assignment 4

Loops

Objectives

The goal of this assignment is to practice with Loops in C++ programs.

Readings

You should be reading chapter 5 of the textbook. The lectures and labs provide additional supporting material.

Instructions

For each **Problem Statement**, follow the steps below:

1. Read the problem statement and clarify the problem.
    a. Break the problem into smaller problems if needed.
2. Determine the IPO.
    a. Determine input, output, intermediate variables, constants, conditions, and repetitions.
    b. Declare the variables and constants (data type + meaningful names).
    c. Work out the problem by hand using typical input values. Determine the range of valid input values.
    d. Determine the process.
3. Write a pseudocode as required.
4. Write a C++ program (use the given filename) that implements the pseudocode.
    a. Add comments where needed. Make sure to use a comments header to reflect the intention of your program and name of the author (you) and the date the program was written.
    b. Test, debug, and execute the program using typical values.

Submit according to the instruction in the "Submission" section.

Problem Statements

1. (grade.cpp) Write a C++ program that receives the grades for an unknown number of students and finds and displays the average and maximum grade. Use a while loop. The input ends with -1. Here is a sample run:

```
Enter the grades (Enter -1 when done): 65 48 83 93 -1
The maximum grade is 93 and the average is 72.25
```

2. (grade_text.cpp) Write a C++ program that receives the grades for 5 students from a text file (the name of file is grading.txt and you create it yourself). The program finds and displays the minimum grade. Use a for loop. Here is a sample run:

```
The values read from the text file: 65 48 83 93 78
The minimum grade is 48
```

3. (guess_number.cpp) Write a **C++ program** that generates a random number between 1 and 200. The program will keep asking the user to input a number and display corresponding message based on the input, telling the user if the guess is bigger or smaller than the generated number or out of range. If the user enters the correct answer, the program will display a message showing how many guesses have been made and exit (if the guess is out of range, it does not count as one guess). Here is a sample run:

```
A number is generated, make a guess: 68
Your guess is too big, make a guess: 20
Your guess is too small, make a guess: 202
Your guess is out of range, make a guess: 44
You got it right! It took 3 times to get the correct answer.
```

4. (prime.cpp) A prime number is a number that cannot be formed by multiplying two smaller numbers (not including 1). For example, 2, 3, 5 are prime numbers but 4 (2*2) and 6 (2*3) are not.

   Write a **C++ program** that receives the start point and end point from user and displays all the prime numbers in this range. The program also receives how many numbers to display per row. **No need to validate**. Print the results in a tabular format. Here is a sample run:

```
Enter the start point: 1
Enter the number of prime numbers: 100
Enter how many numbers to display per row: 10
The prime numbers from 1 to 100 are:
2    3    5    7    11   13   17   19   23   29
31   37   41   43   47   53   59   61   67   71
73   79   83   89   97
```

5. (count_char.cpp) Write a **C++ program** that prompts the user to enter a string and displays the number of the digits, uppercase letters, lowercase letters, and spaces in the string. Here is a sample run:

```
Enter a string: CPSC 1155 is Fun!
The number of digits is 4
The number of uppercase letters is 5
The number of lowercase letters is 4
The number of spaces is 3
```

6. (substring.cpp) Write a **C++ program** that reads two strings and determines if the shorter string is a substring of the longer one. The program also determines the position that substring starts in the longer string.

   Here is a sample run:

```
Input a string: textbook
Input another string: book
"book" is a substring of "textbook" starting at letter 5

Input a string: txt
Input another string: textbook
"txt" is not a substring of "textbook"
```

7. (factors_of_integer.cpp) Write a **pseudocode** and **C++ program** that reads an integer and displays all its smallest factors in increasing order. For example, if the input is 120, the output should be: 2, 2, 2, 3, 5.

8. (display_pattern.cpp) Write a **C++ program** that displays a pattern of stars and numbers. The program reads a number in the range 1 to 10 to specify the number of columns in the pattern. You may use `cout` statements that print either a single asterisk (*) or a single digit. Maximize your use of repetition (with nested `for` statements) and minimize the number of `cout` statements. Your program should then display a pattern of the appropriate size. [Hint: You could use more than 1 nested loops.]

   Here are two samples:

```
With 3 as the input:
1**
*2*
**3
**3
*2*
1**
```

```
With 4 as the input:
1***
*2**
**3*
***4
***4
**3*
*2**
1***
```

Submission

Submit a zip folder named as yourName_Assign4.zip to Brightspace. This folder should consist of the C++ codes in individual .cpp files and one pseudocode.txt file with your **pseudocode** (question 7). Please name each cpp file exactly as the instructed names at the beginning of each question.

Please make sure that all your .cpp files compile and run properly before submission. Your file must run properly in order to receive full marks.

Marking Scheme

There are 10 marks for each question with the following details:

Question 1: grade.cpp

- 7 for getting maximum or average correctly
- 3 for getting both correctly

Question 2: grade_text.cpp

- 10 for correct solution

Question 3: guess_number.cpp

- 2 for checking boundary
- 6 for correct solution
- 2 for times counting

Question 4: prime.cpp

- 6 for correct solution
- 4 for correct display format

Question 5: count_char.cpp

- 10 for correct solution

Question 6: substring.cpp

- 10 for correct solution

Question 7: factors_of_integer.cpp

- 4 for pseudocode
- 6 for correct solution

Question 8: display_pattern.cpp

- 10 for correct solution